

OIDC Advanced Syntax for Claims (ASC)

Transformed Claims & Selective Abort/Omit

Daniel Fett, 2022-04-27

OIDC Advanced Syntax for Claims (ASC)

Defines extensions for OIDC around **requesting and receiving Claims**

But... why?

- Fine-grained control over data delivery
 - ... for privacy
 - ... for billing
 - Clients need to be able to define **what they want**,
 - and **don't pay** for data useless to them.
- Handling of complex Claims
 - OpenID Connect Core: **2** levels (root+1, e.g., in 'address')
 - OpenID for Identity Assurance: **5+** levels

OIDC Advanced Syntax for Claims (ASC)

- No dependency on OpenID Connect for Identity Assurance (OIDC4IA), but:
 - Requirements derived from eKYC work
 - Special provisions for combination cases with OIDC4IA
- Two independent extensions:
 - ASC/SAO: Selective Abort/Omit
 - ASC/TC: Transformed Claims (among others, for age verification)

ASC/SAO
Selective Abort/Omit

“the feature you expected
essential: true to be”

Selective Abort/Omit

Cases covered:

- **if_unavailable**
- **if_different**

Actions:

- **abort**
- **omit**
(omit this Claim)
- **omit_set**
(omit all Claims with
omit_set)
- **omit_verified_claims**

```
{
  "id_token": {
    "phone_number": {
      "if_unavailable": "abort"
    },
    "custom_paid_claim": {
      "if_unavailable": "omit_set"
    },
    "verified_claims": {
      "verification": {
        "trust_framework": {
          "value": "de_aml",
          "if_different": "abort"
        },
        "verification_process": {
          "if_unavailable": "omit_verified_claims"
        }
      },
      "claims": {
        "given_name": null,
        "family_name": null,
        "place_of_birth": {
          "if_unavailable": "omit_set"
        }
      }
    }
  }
}
```

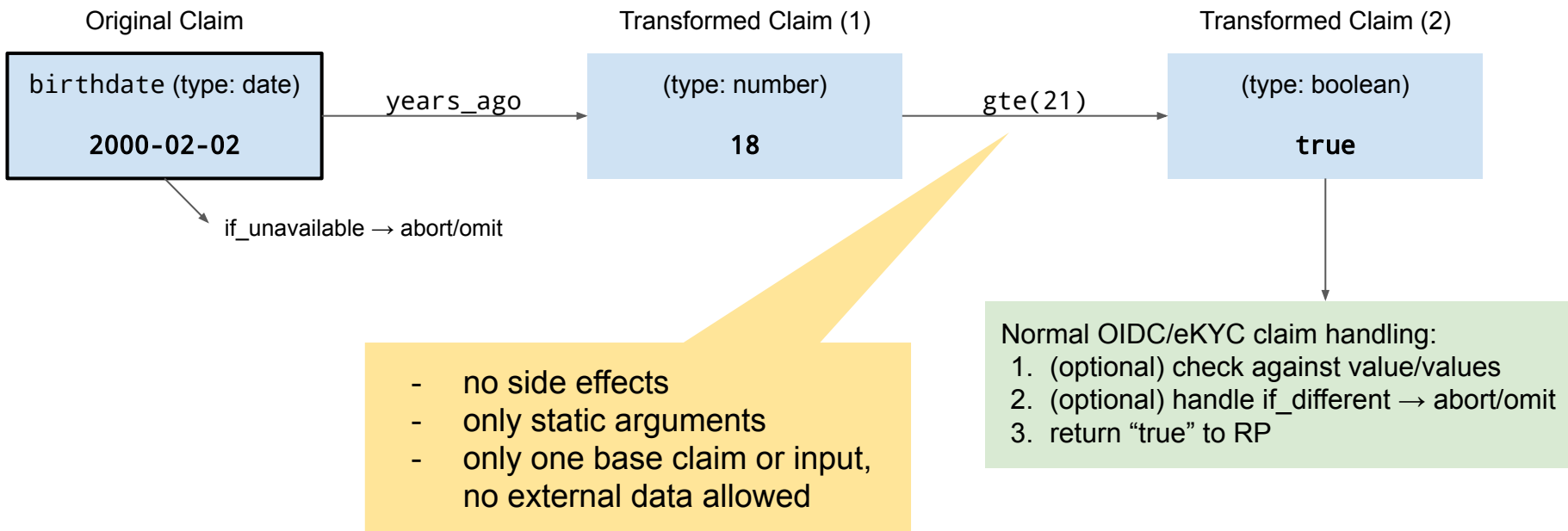
ASC/TC
Transformed Claims

Use Cases

- Age Verification:
 - Above 16? Above 18? Above 21? Under 99?
- Partial matching:
 - E-Mail ends with '@company.com'
 - ZIP code is '90210'
 - address/country is not empty
 - Nationalities contains 'JPN'
- Data minimization:
 - Return only address/country instead of address

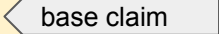
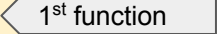
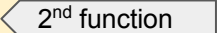
Idea

Claims values can be transformed using a small set of functions before any further evaluation is performed:



Example: Age Verification

Definition

```
claims=  
{  
  "transformed_claims": {  
    "above_18": {  
      "claim": "birthdate",   
      "fn": [  
        "years_ago",   
        ["gte", 18]   
      ]  
    }  
  },  
  "id_token": {  
    "given_name": null,  
    "family_name": null,  
    ":above_18": null  
  }  
}
```

Use - Prefix ':'

Response:

```
{  
  ...  
  "given_name": "Max",  
  "family_name": "Mustermann",  
  ":above_18": true,  
  ...  
}
```

Simple, Self-Contained Functions

- **years_ago(optional date ReferenceDate): date → number**
Takes a date (or datetime), calculates the number of years since the date. Optionally, a reference date is given.
- **gt(number Threshold): number → boolean**
lt(number Threshold): number → boolean
Evaluate whether a number is above/below a certain threshold.
- **any(): array of booleans → boolean**
all(): array of booleans → boolean
none(): array of booleans → boolean
Evaluate whether, in an array of booleans, any, all, or none of the values are “true”.
- **eq(any Compare): any → boolean**
Evaluates equality - useful in combination with any/all/none for arrays.
- **get(string Key): JSON object → any**
Access the key of a JSON object; returns the value.
- **match(string Regex): string → bool**
Match a string against a regular expression. (Todo: Define a regex dialect and/or subset to support.)

Example: Partial Matching

```
claims=  
{  
  "transformed_claims": {  
    "company_email": {  
      "claim": "email",  
      "fn": [  
        ["match", "@company\\.com$"] ← 1st function  
      ]  
    },  
    "nationality_usa": {  
      "claim": "nationalities",  
      "fn": [  
        ["eq", "USA"], ← 1st function  
        "any" ← 2nd function  
      ]  
    }  
  },  
  "id_token": {  
    ...  
    ":company_email": { "value": true, "if_different": "abort" },  
    "email_verified": { "value": true, "if_different": "abort" },  
    "verified_claims": {  
      "claims": {  
        ":nationality_usa": { "value": true, "if_different": "abort" }  
      },  
      "verification": { "trust_framework": null }  
    }  
  }  
}
```

Simplifying Common Use Cases

- OPs can opt to support only a limited subset of functions:

OP Metadata:

```
"transformed_claims_functions_supported": ["years_ago", "gte"]
```

- OPs can provide Predefined Transformed Claims (PTC):

OP Metadata:

```
"transformed_claims_predefined": {  
  "above_18": {  
    "claim": "birthdate",  
    "fn": [  
      "years_ago",  
      ["gte", 18]  
    ]  
  }  
}
```

- OPs can limit support to PTCs only:

OP Metadata:

```
"transformed_claims_restricted": true,
```

Example: Age Verification with PTC

:: indicates PTC

```
claims=  
{  
  "id_token": {  
    "given_name": null,  
    "family_name": null,  
    "::above_18": null  
  }  
}
```

```
Response:  
{  
  ...  
  "given_name": "Max",  
  "family_name": "Mustermann",  
  "::above_18": true,  
  ...  
}
```

With PTCs, simple use cases can be handled with **minimal implementation overhead**, both for OP and RP.

The PTC is handled just like any other custom Claim, but has a **precisely-defined meaning**.

UX Considerations

- For PTCs, OPs can trivially show a meaningful consent prompt
- For Custom TCs, OPs can try to match patterns:
 - e.g. `birthdate / years_ago / gte(x)` → Consent: “RP wants to know whether you are `x` years old or above”.
- Safe fallback:
 - Show consent to release of full Claim (“wants to know your birth date”)
 - → safe overapproximation because:
 - no side effects,
 - no expressions over multiple Claims,
 - no dynamic arguments

Compatibility Considerations

- New element “transformed_claims” will be ignored by non-supporting OPs
- Transformed Claims will be ignored by non-supporting OPs
- RPs can check OP support in metadata
- Ecosystems can define custom functions
- Can be used with and without ASC/SAO.

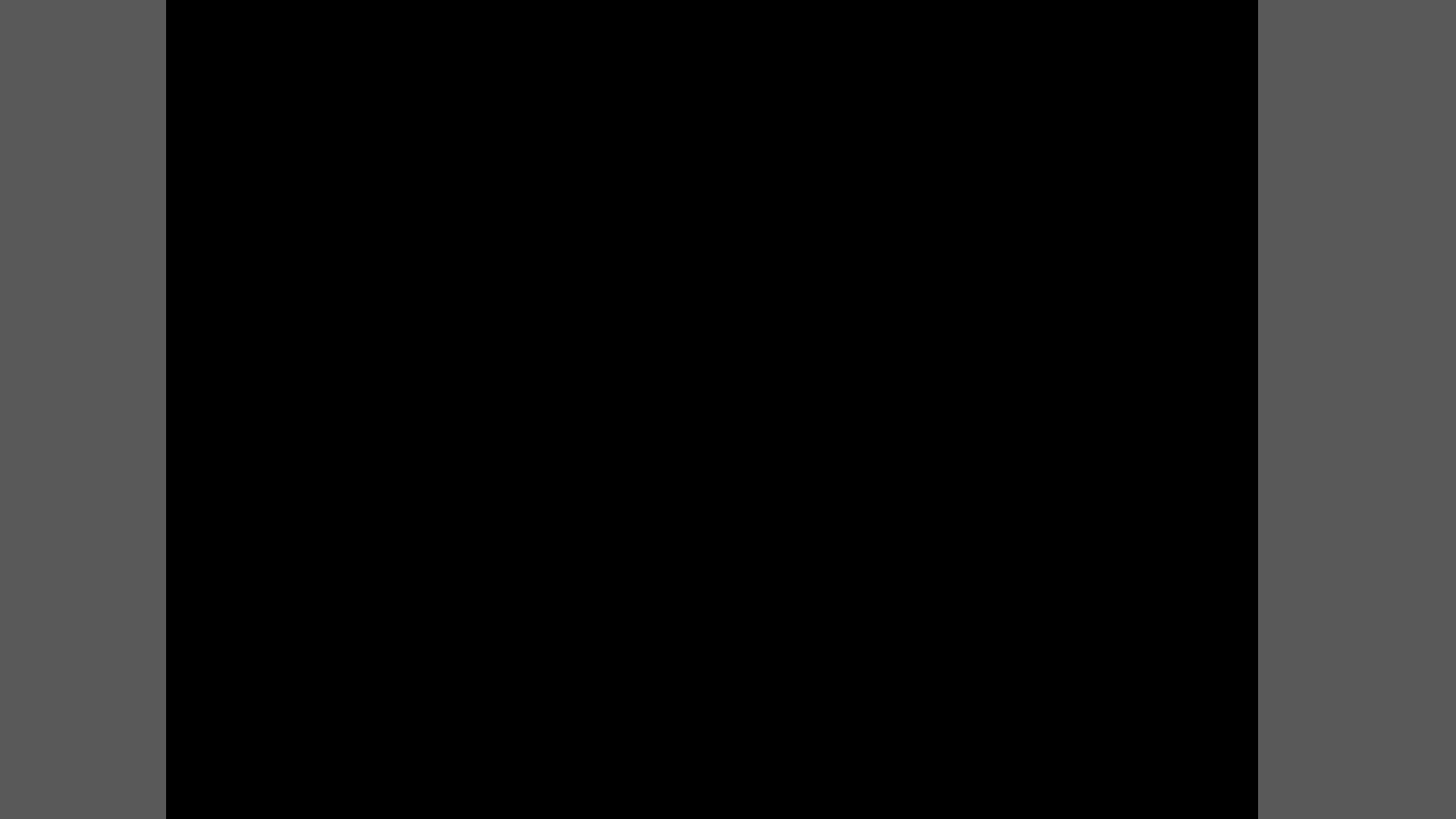
Demo: Authlete by Takahiko Kawasaki

```
{
  "transformed_claims": {
    "age": {
      "claim": "birthdate",
      "fn": [
        "years_ago"
      ]
    },
    "18_or_over": {
      "claim": "birthdate",
      "fn": [
        "years_ago",
        [ "gte", 18 ]
      ]
    },
    "below_18": {
      "claim": "birthdate",
      "fn": [
        "years_ago",
        [ "lt", 18 ]
      ]
    },
    "country": {
      "claim": "address",
      "fn": [
        [ "get", "country" ]
      ]
    }
  },
```

```
    "country_germany": {
      "claim": "address",
      "fn": [
        [ "get", "country" ],
        [ "match", "^[Gg]ermany$" ]
      ]
    },
    "nationality_usa": {
      "claim": "nationalities",
      "fn": [
        [ "eq", "USA" ],
        "any"
      ]
    },
    "nationality_japan": {
      "claim": "nationalities",
      "fn": [
        [ "eq", "JPN" ],
        "any"
      ]
    }
  },
  "id_token": { ( ... ) }
}
```

```
{
  "kid": "demo",
  "alg": "ES256"
}
```

```
{
  ":18_or_over": true,
  "sub": "1003",
  "address": {
    "locality": "Augsburg",
    "region": "Bavaria",
    "country": "Germany"
  },
  "birthdate": "1956-01-28",
  ":below_18": false,
  ":country": "Germany",
  "iss": "https://authlete.com",
  ":country_germany": true,
  "nationalities": [
    "USA",
    "DEU"
  ],
  "nonce": "mynonce",
  ":nationality_japan": false,
  "aud": [
    "8354507233"
  ],
  ":nationality_usa": true,
  "auth_time": 1638113730,
  "exp": 1638200130,
  "iat": 1638113730,
  ":age": 65
}
```



Current Status

- Draft in the OpenID Foundation eKYC & IDA working group
- Transformed Claims: No open issues, first implementation
- Selective Abort/Omit:
 - Main issue: Execution order of rules and recursive effects
 - Expect some changes